Airline Scheduling

Greg Carroll and Vicky Kennard

Airline Scheduling

These notes accompany the video on Airline Scheduling.

Click on this link to watch the video: **Airline Scheduling Video**

Introduction

This module provides background material to the AMSI video 'Airline Scheduling'. The video provides a stimulus for the study of networks and linear programming which form part of the ACARA Senior General Maths Course. All States include this material in their Senior Mathematics courses.

In the video the field of mathematics called 'Operations Research' is discussed. This is an umbrella term for several decision-making methods studied in ACARA General Mathematics, i.e. networks, trees, minimum connector problems, critical path analysis, network flow, assignment problems and linear programming.

Throughout this module there are short exercises that can be used to check on your understanding as you proceed.

Flight Scheduling

In the video you see the complexity involved in running a modern airline business. Airlines price and advertise flights based upon historical data for the number of paying passengers wanting to fly between two airports. This is just the beginning of the multifaceted task of assigning aircraft, crew, fuel, catering and luggage as well as reserving the departure and arrival gates.

Scheduling flights also involves the dimension of time. The availability of runways, gates, crew etc. are all time dependent. No single flight operates without consideration of where that plane will fly next and who the next crew will be. International airlines may also need to consider the language spoken by the crew and at the destination.

There are many areas of operational research that are relevant to the challenges that airlines face. These include:

- scheduling – having the right equipment, people etc. in the right place at the right time

- optimizing – using equipment and people efficiently to minimize waste/cost and maximize profit

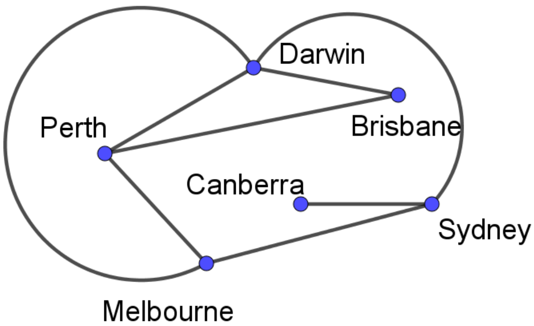
- assignment/allocation – putting the best people, equipment, contracts together to minimize cost and increase efficiency.

Some of the methods used in this module may be familiar to you from your studies; others may be new. The following pages will break down this complex task and introduce the language of networking and Operations Research, through applying these methods to the airline industry.

Simple Graphs

Networking

Consider the following diagram which shows six cities and some of the flights between them.



We can see that, apart from Canberra, each city has at least 2 flights leaving from it.

In mathematical terms, each of the cities is called a **node** or **vertex** (plural is vertices) and the links between them are **arcs** or **edges**. The number of arcs leaving a node is called the **degree** of the node.

Exercise 1

a Name Perth's connections.

b Who is Canberra's only connection?

c What is the degree of Melbourne's node?

Another way of representing this information is called an **adjacency matrix**. This lists the number of ways a node can be reached from any other node.

	P	B	C	D	S	M
P	0	1	0	1	0	1
B	1	0	0	1	0	0
C	0	0	0	0	1	0
D	1	1	0	0	1	1
S	0	0	1	1	0	1
M	1	0	0	1	1	0

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Exercise 2

- a Why is the diagonal from the top left to the bottom right all zeroes?
- b What does the total of each row or column represent?
- c Which city has the most connections?

Drawing the network from the matrix

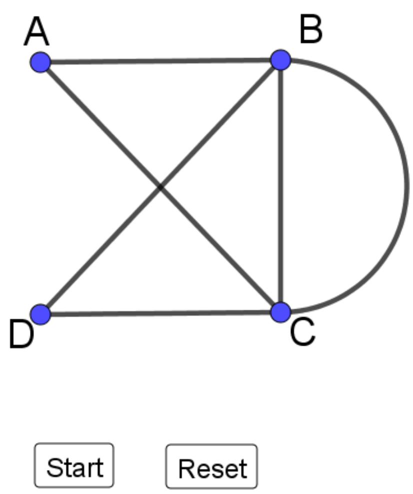
Often there is more than one way to draw the network from the matrix, but as long as each has the correct connections between nodes they are equally correct; these are called **isomorphic graphs**.

For example, consider the adjacency matrix below.

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

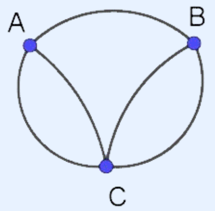
The graph below is drawn from the adjacency matrix above. If you click on the link you will see an animation of how this graph can be redrawn so that no edges cross. The resulting graph is now called **planar**. Not all graphs can be drawn as planar but it is better to do so if you can.

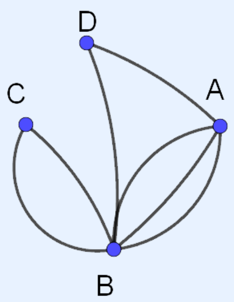
Click on the diagram to see it animated. (This will open the link in a browser.)



Exercise 3

a Write the adjacency matrix for the following networks.





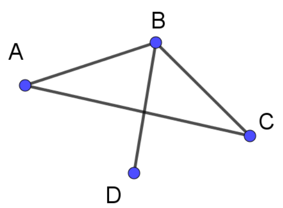
b Draw a network diagram to represent these adjacency matrices. Make your graphs planar if possible.

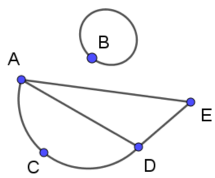
$$\begin{bmatrix} 0 & 3 & 2 \\ 3 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 2 & 0 & 2 \\ 2 & 0 & 4 & 2 \\ 0 & 4 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{bmatrix}$$

When we are looking at flight scheduling, the edges are the flight paths and the nodes are the airports. Keep in mind that the availability of any node (airport) is time dependent, as more than a single plane uses the airport.

A simple graph is **connected** if every node can be reached from every other node either directly or via a sequence of edges.

Looking at the graphs below, the graph on the left is connected because every node can be reached from any other node. The graph on the right is disconnected because the node B cannot be reached from all of the other nodes. (In is case, it cannot be reached from *any* of the other nodes!) However, B does contain an arc beginning and ending at B, and this is called a **loop**. You could think of this as a scenic helicopter flight, that takes off and lands at the same airport. Even though there is only one path joining B to B, the loop still adds 2 to the degree of B.





A Bit of History

In the video, Euler (pronounced 'oiler') and the famous 'Bridges of Konigsberg'problem were mentioned as the origin of Operations Research. Although he and other mathematicians were looking at these problems many years ago, the real beginnings of this modern field of mathematics was around the time of the Second World War, by the military.

During the war there was a need to move large numbers of personnel and equipment across land and sea. This gave rise to complex logistical problems that required methods for finding optimal solutions, quickly.

After the war other areas of commerce and industry recognised the usefulness of these algorithms and started to employ them. This new field of mathematics is now called Operations Research. Many problems involve multiple laborious and repetitive calculations, and these have only become practical with the introduction of computers.

Other famous problems in this field include the Travelling Salesman and the Postman problems. These illustrate two of the main types of scheduling – the need to visit every node at least once **or** to travel along every edge at least once. The table below details two well known solutions to these problems.

Table 1: **"Every Node or Every Road"**

Euler Circuit	Uses every edge exactly once Starts and finishes at the same node Possible if the degree of **every** node is **even**
Euler Path	Uses every edge exactly once Starts and finishes at different nodes Possible if there are **only** two **odd** nodes (the start and the finish)
Hamiltonian Circuit	Visits every node exactly once. (Start/Finish node is an exception) Starts and finishes at the same node
Hamiltonian Path	Visits every node exactly once Starts and finishes at different nodes

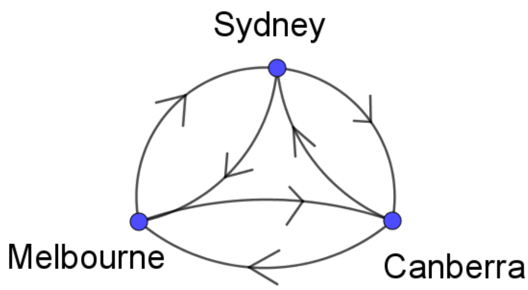
There have been 11 Nobel Prize winners in Economics (because there is no Nobel prize for Mathematics) that have used ideas from operations research. The Daniel H. Wagner Prize is awarded, each year, for a paper and presentation that describe a real-world, successful application of operations research or advanced analytics. This prize is given by the Institute for Operations Research and the Management Sciences (INFORMS), the largest society of professionals in the field of operations research, management science, and advanced analytics.

See link to INFORMS website.

Directed Networks

Consider the following network of flight paths from Sydney to Melbourne. Since we do not want the planes travelling *from* Sydney to crash head on with those travelling *to* Sydney, there is more than one flight path. In fact, with multiple airlines travelling from multiple destinations there are many paths in and out of every major airport. Flight paths exist in 3 dimensions; there is a difference in height for different paths.

This diagram shows one flight in and out of each of the three airports. Note the arrows on the edges. These restrict the direction in which you can travel along that edge. This is called a **directed graph** or **digraph**.



This diagram shows a situation where it is possible to fly directly from one airport to another, but this is true only for the major airports. For destinations with a smaller number of potential passengers, the passengers meet at a nearby major airport, called a hub, before flying to the smaller destination.

In Figure 2 showing the flight paths in Western Australia, you can see that passengers needing to fly to the town of Newman must all fly to Perth first. When leaving Newman, to go anywhere, you must first fly to Perth.

When you make a booking on a website, the computer uses an algorithm to find a route between two airports, or nodes. Many computer-booking systems allow you to specify if you want your route to be the shortest in terms of distance, time, number of stops or for the least cost. The **optimal** (best) route may vary if you change these constraints.

An interesting application of directed networks is UPS's directive to its delivery drivers to plan routes that only ever use right turns (they drive on the other side of the road in the USA). An individual route may seem to be longer, but UPS has calculated that this saves significant fuel (over 37 million litres) and allows them to deliver 350,000 more packages every year. It also, counter-intuitively, saves on distance travelled (over 45 million km each year).

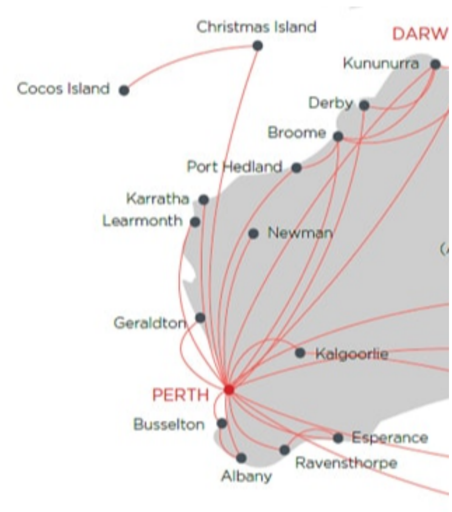


Figure 2: Virgin Australia domestic network

What might appear to be the obvious, sensible, solution is not always the optimal one.

See link to UPS story

Not all algorithms deliver the **optimal** (best) solution every time. They give ‘a’ solution, but not the necessarily the optimal one. In theory, computer algorithms could be used to generate all possible solutions and then search for the best one, but in practice this is infeasible due to the huge amounts of time it would take for such a method to run. Mathematicians are always searching for fast algorithms which can either give the optimal solution, or that are guaranteed to be close to optimal.

Shortest path analysis – Dijkstra’s algorithm

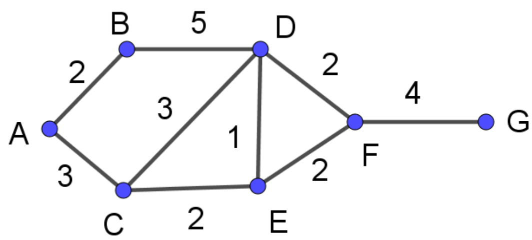


Figure 3: A network from A to G

Consider a passenger wanting to get from A to G and there is no direct flight. A variety of stopovers and routes are possible as shown in this network. It may be possible to find the shortest path by trial and error but how would we ever be sure it was the shortest path? Dijkstra’s algorithm for the shortest path will provide a reliable correct answer every time.

The numbers, or **weights**, on the edges represent time, cost, distance etc. depending on the problem. In this example the numbers represent hours of travel between the nodes. The nodes themselves represent places such as home, office, airports, hotels etc. Diagrams and tables usually use letters, rather than names, for the nodes. This is done to keep the diagram simple.

Dijkstra's algorithm looks for the shortest path from a given vertex to another given vertex. It does not have to include all the intervening vertices. The best way to learn about how Dijkstra's algorithm works is to watch this **YouTube clip**.

Example

Suppose we want to find the shortest path from A to G in Figure 3. Following the algorithm in the video we get the table below:

Vertex	Shortest distance from start	Previous vertex
A	**0**	
B	∞, **2**	**A**
C	∞, **3**	**A**
D	∞, 7, **6**	B, **C**
E	∞, **5**	**C**
F	∞, **7**	**E**
G	∞, **11**	**F**

The shortest path is **A – C – E – F – G** (Note: this path does not include the node D)

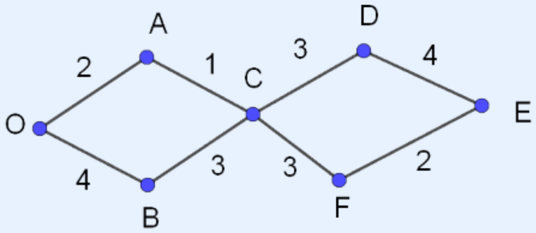
In this context the passenger would be travelling for 11 hours to reach their destination.

When you use your GPS or online mapping app such as Google Maps, there is an algorithm behind the scenes – possibly a more complex form of Dijkstra's method – finding the shortest route to your destination. These algorithms will also consider traffic conditions and your own preferences for travel, such as using toll roads. Another complication are restricted, or one-way, roads. These are called **directed paths** and will be discussed in the next section.

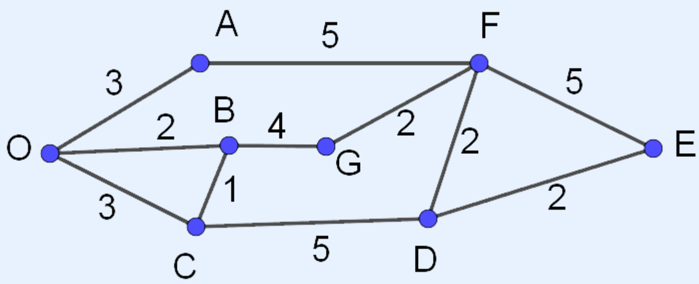
Exercise 4

Use Djikstra's algorithm to find the shortest path from O to E in each of these networks.

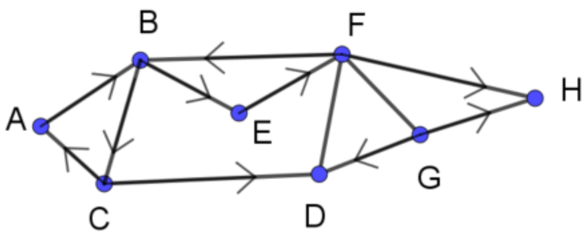
a



b



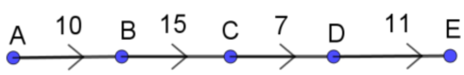
Directed paths



To find a path from A to H we can only travel in the direction indicated by the arrow on each edge.

A–B–C–D–F–H is an acceptable path, and so is **A–B–E–F–H**.

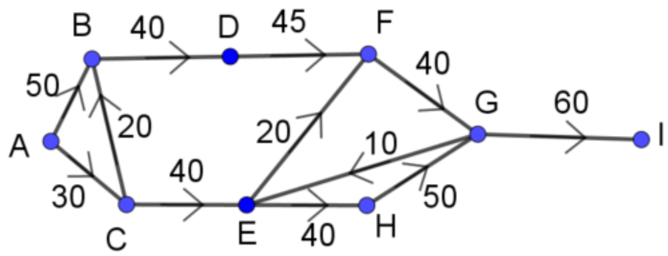
Network Flow



We can use numbers above the edges to represent capacity or time. In this diagram the numbers represent the maximum number of flights that can be handled per hour between five airports. The maximum **flow** that can be achieved by this system is found by looking for the smallest number.

Since there is only room for 7 flights between C and D, the maximum flow is 7.

Consider this more complicated system.

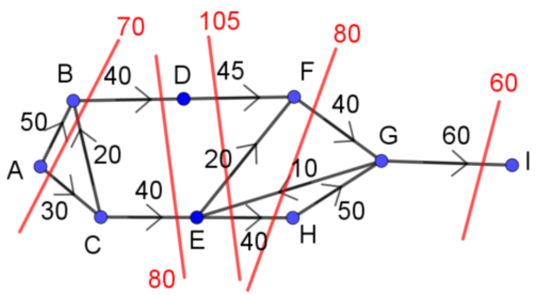


Here the figures represent the maximum number of cars that can use each edge per hour. What is the maximum number of cars per hour that can reach I from A?

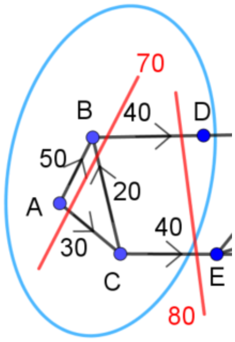
We can see that although theoretically 80 cars could leave A per hour (30 to A and 50 to B) there is a maximum of 60 that can travel from G to I. Is it possible that there is an even smaller bottleneck somewhere else in this network?

We could use common sense to see if there is a point in the network with a capacity of less than the 60.

A more efficient technique is to take **cuts** through the network looking for the smallest capacity. In this network the minimum cut is 60, so that is the maximum capacity.



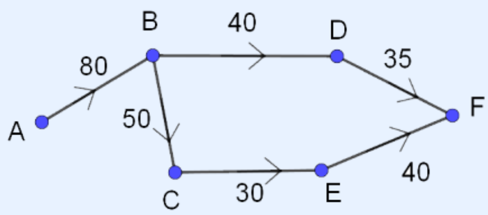
Note: Some of the cuts seem to have more capacity than the figure indicates but this is because we only include the routes travelling from the left-hand side of the cut to the right, the same direction as the traffic needs to travel. We ignore the edges that are directed in the opposite direction.

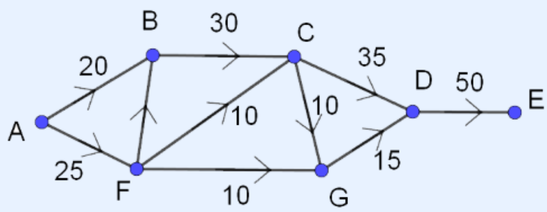


For example, in this part of the diagram, three edges are cut, (AC, BD, CB). The cut AB (30) and the cut BD (40) are in the direction of travel, and cut CB (20) goes in the reverse direction so is not included.

Exercise 5

a Find the maximum flow for each of these network diagrams.





b Which road/s in the first network could be upgraded to improve the capacity of the network?

Minimum Spanning Trees (MST)

A **tree** (or **spanning tree**) is a collection of edges within a graph which connects all the nodes together, without creating any loops (or cycles). A **minimum spanning tree** is the tree which has the smallest possible total weight, i.e. the sum of the edges is the smallest it can be.

Consider this problem: a new airport has four terminals, A, B, C and D, to be connected to the same runway, F. To save money the construction company wants to build a tree that connects all terminals to the runway at the minimum cost.

There are two algorithms we could use to search for a solution:

- **Prim's algorithm** starts with a single node. At each step it finds all the edges that connect the tree to nodes that are not yet in the tree, finds the edge of smallest weight, and adds this edge to the tree. It continues until all nodes are in the tree.
- **Kruskal's algorithm** starts from the shortest edge (the edge with the least weight). At each step it searches for the next-shortest edge that is not in the tree. If adding it to the tree would form a cycle then it is discarded, otherwise it is added to the tree. It continues until all nodes are in the tree.

Applying these algorithms to the same networks may lead to different paths, especially if there are many edges with the same weights. Even where the end result is the same, the edges may be added in different orders. Using a computer enables us to find all possible minimum spanning trees, though all have the same total weight so none is better than any other.

Click here to see an animation of Prim's and Kruskal's algorithms

These algorithms are described in computer terms as greedy algorithms. This is because they choose the best option at each step in the process. Prim's and Kruskal's algorithms are guaranteed to find correct minimum spanning trees but, in general, greedy algorithms may not give the optimal solution to a problem.

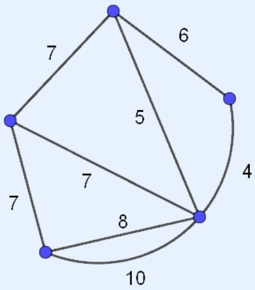
Exercise 6

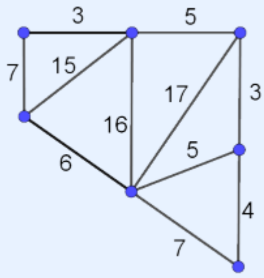
For each of the graphs below:

a Find a minimal spanning tree.

b State whether you used Prim's or Kruskal's algorithm.

c Can you find more than one solution with the same weight?





Critical Path Analysis (CPA)

Critical Path Analysis, often used in project management, is a method for finding the optimal path for a series of activities. The weights on the edges represent time, cost, distance etc. The aim of CPA is to identify those activities that have to be completed, in a specific order, called *critical activities*, and how other activities can then be 'slotted' in to fit.

For example, we could use CPA to find the shortest time for our morning routine. Some activities have to be completed in a certain order (e.g. we cannot get dressed until after we have showered) while others can be done simultaneously (e.g. we can set the toaster going while we put our socks on).

In this kind of project network, the **float** or **slack** of an activity is the amount of time that the task can be delayed without affecting the total completion time of the project. It is a measure of the flexibility in a project.

The two video clips below explain how to conduct a CPA using forward and backwards passes and how to identify the float/slack times for the non-critical activities.

- **This video explains how to find the critical path using forward and backward passes**
- **This video shows how to calculate the float/slack times for non-critical activities**

We have created a Geogebra animation to show a simple example of forwards and backwards scanning as well as how to construct a network diagram from a table of precedences. Click here to see this animated CPA example in Geogebra. To start or pause the animation, click the “On/Off” tick box.

Exercise 7

Using the data in the table below, complete the following:

a Draw a network diagram.

b Perform a forward pass to determine the earliest completion time for the project.

c Perform a backward pass and determine the critical path.

d What are the float times for the non-critical activities?

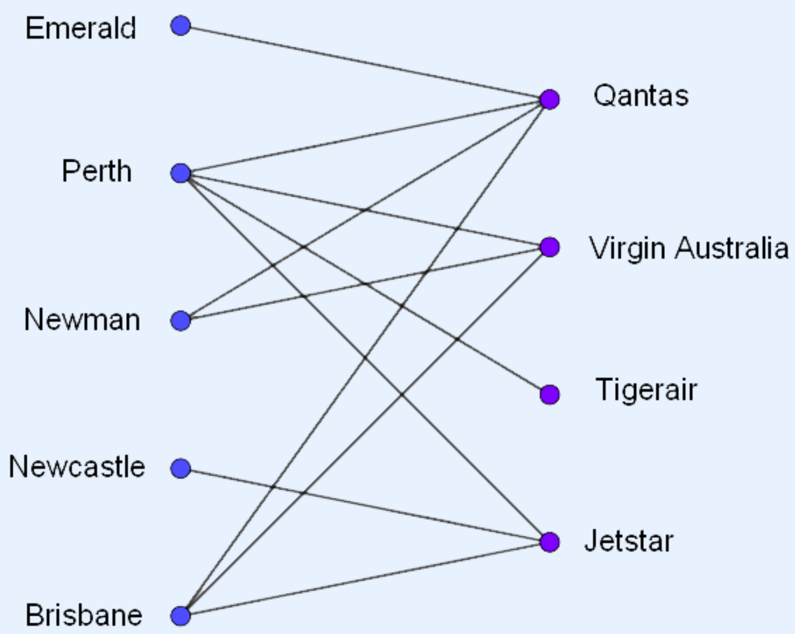
Activity	Predecessor	Time (hours)
A	–	1
B	A	2
C	B	1
D	–	3
E	–	3.5
F	D	2.5
G	C, E, F	1.5
H	G	1
J	H, K	2
K	C. E, F	2

Bipartite Graphs

A **bipartite graph** is a graph between two **distinct** sets of vertices. These types of graph are used to model the relationships between two groups and are often used to allocate resources; for example, crews to flights. These are often called *matching problems*.

Exercise 8

Bipartite graphs can be used to show which airlines fly to certain destinations.



a Which airlines fly to Brisbane?

b Which destinations does Virgin Australia fly to?

c Which airline flies to only one of the listed destinations?

Exercise 9

Four flight crews were asked for their preferences regarding their next destination. Draw a bipartite graph to show this data.

Jane	Canberra, Sydney
Peter	Sydney, Melbourne
Malik	Canberra
Leah	Sydney, Perth
Joe	Perth, Canberra

Warehousing

Consider the following airport catering example. Four different planes must be supplied with meals before departure, by four different catering companies. The costs for each company, C1, C2, C3 and C4, to supply each of the flights, F1, F2, F3 and F4, are in the table below (which we will also call a matrix). Each company will supply a single plane. The airline company wants to minimise costs. The task is to find the cheapest way to cater for each plane.

	F1	F2	F3	F4
C1	82	83	69	92
C2	77	37	49	92
C3	11	69	5	86
C4	8	9	98	23

For a relatively small example like this, it is tempting to use trial and error. We could start by allocating the cheapest prices first, i.e. C3 to F3 (5) and C4 to F1 (8). This would allow us to allocate the combination of C1 and C2 to F2 and F4 as cheaply as possible: C1 to F4 (92) and C2 to F2 (37).

This gives a total cost of $5+8+92+37=142$.

Is this the minimum? How do we know?

The Hungarian Algorithm

The Hungarian algorithm was designed to solve allocation, or assignment, problems and produces the optimal solution. The algorithm was developed and published in 1955 by Harold Kuhn, who gave it the name "Hungarian method" because the algorithm was largely based on the earlier work of two Hungarian mathematicians: Dénes Kőnig and Jenő Egerváry.

The following example is calculated by hand; for more complex problems, computer algorithms are used.

Step 1: To simplify the matrix start by subtracting the **minimum** figure in each row from all members of that row.

If we start with the matrix we were given before:

	F1	F2	F3	F4
C1	82	83	69	92
C2	77	37	49	92
C3	11	69	5	86
C4	8	9	98	23

then after Step 1 it becomes

	F1	F2	F3	F4	
C1	13	14	0	23	(–69)
C2	40	0	12	55	(–37)
C3	6	64	0	81	(–5)
C4	0	1	90	15	(–8)

Step 2: Subtract the column minima from each of the members of that column. Note: the first three columns already contain a zero so there is no change.

	F1	F2	F3	F4
C1	13	14	0	8
C2	40	0	12	40
C3	6	64	0	66
C4	0	1	90	0
				(–15)

Step 3: Cover all of the zeroes in the matrix using the **minimum** number of straight lines. If the number of lines is less than the size of the matrix, go to Step 4; otherwise, stop.

	F1	F2	F3	F4
C1	13	14	0	8
C2	40	0	12	40
C3	6	64	0	66
C4	0	1	90	0

All the zeroes can be covered using three lines. Since this is less than the size of the matrix, four, we need to proceed to the next step.

Step 4: Find the smallest number not covered by one of the lines drawn in Step 3. Subtract this from all the uncovered elements, and add it to those elements covered twice.

The smallest uncovered number is 6, so we do the following calculation:

	F1	F2	F3	F4
C1	$13-6=7$	$14-6=8$	0	$8-6=2$
C2	40	0	$12+6=18$	40
C3	$6-6=0$	$64-6=58$	0	$66-6=60$
C4	0	1	$90+6=96$	0

This becomes:

	F1	F2	F3	F4
C1	13	14	0	8
C2	40	0	18	40
C3	0	58	0	60
C4	0	1	96	0

It now requires four lines to cover all the zeroes, the same as the matrix size, so the algorithm is complete.

We can now assign the catering jobs by allocating tasks where a single 0 exists in a row or column.

In our example, this means: C1 to F3, C2 to F2 and C4 to F4. This leaves C3 for F1.

Returning to the original matrix gives the following allocation:

	F1	F2	F3	F4
C1	82	83	69	92
C2	77	37	49	92
C3	11	69	5	86
C4	8	9	98	23

with the total cost being $69+37+11+23=140$.

Trial and error gave us an answer of 142, so using the Hungarian algorithm has improved the result.

Exercise 10

The costs of catering for flights A, B, C and D from warehouses 1, 2, 3 and 4 are shown in this table.

	A	B	C	D
W1	70	64	62	90
W2	77	37	8	98
W3	35	24	25	88
W4	86	30	100	35

Use the Hungarian algorithm to find which warehouse should supply each plane to minimise costs.

Linear programming (LP)

The airline industry uses the linear programming technique to maximise profits and minimise expenses. This method is used when the problem involves optimising a linear equation or a system of linear inequalities.

As an example of such a problem, we will consider a single flight with the company offering both economy and business class seats. The aircraft has a seating area capacity of $200\,\mathrm{m}^2$. Each economy seat requires $0.8\,\mathrm{m}^2$ and each business class seat $2\,\mathrm{m}^2$. The airline charges \$300 for each economy seat and \$720 for each business class seat. If there is a minimum of 10 business class seats required and we assume the plane is fully booked, what is the ideal number of each class of passengers to maximise income?

The solution proceeds as follows.

Step 1: Define variables:

$e =$ the number of economy passengers

$b =$ the number of business passengers

Step 2: Set the constraints

$0 \le e \le 225$ (since at least 10 passengers are business class, there is $180\,\mathrm{m}^2$

remaining for economy passengers, so the maximum number is $180/0.8 = 225$)

$10 \le b \le 100$ (number of business class passengers)

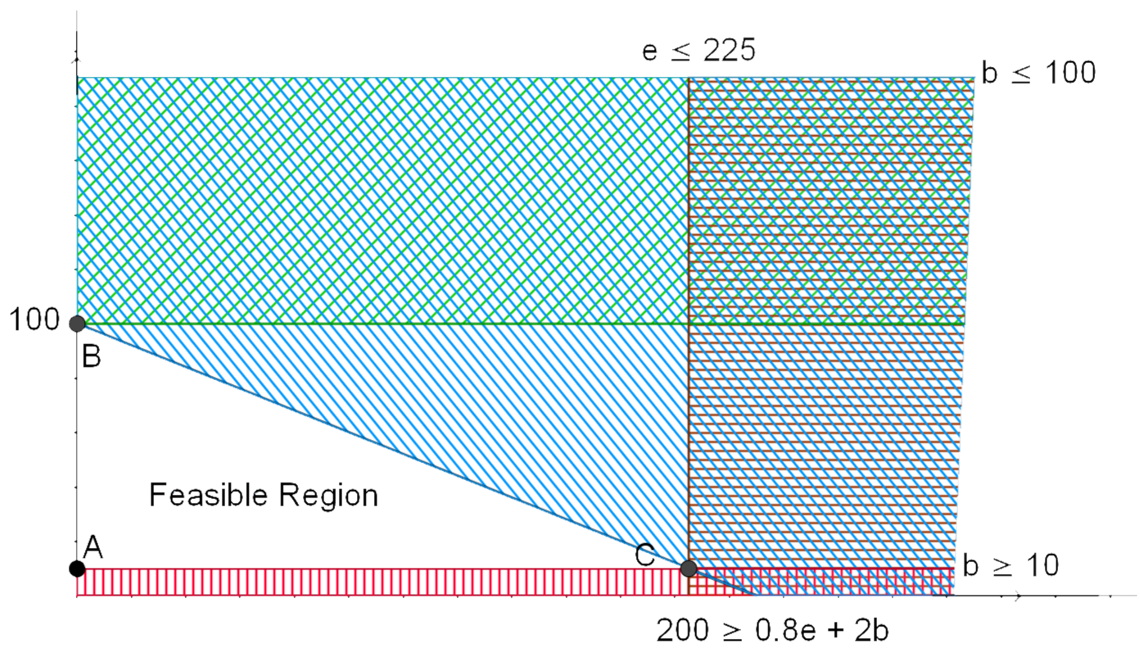
$200 \geq 0.8e+2b$ (total area taken up by passengers cannot exceed the seating capacity)

Step 3: State the objective function: (what we are trying to maximise, or minimise)

In this case we are trying to maximise income, which is \$300 per economy passenger and \$720 per business passenger. Our objective function is therefore

$$P(MAX) = 300e + 720b.$$

Graphing these constraints produces the feasible area with vertices A, B and C, as shown below. The feasible area is the area inside which all the inequalities are satisfied.



The maximum, or minimum, always lies on a vertex (corner) of the feasible region. We test each of the points with the objective function to find which one provides the maximum output:

$A = (0, 10); P = 300(0) + 720(10) = 7200$

$B = (0, 100); P = 300(0) + 720(100) = 72000$

$C = (225, 10); P = 300(225) + 720(10) = 74700$

Therefore the maximum income of \$74 700 is achieved by having 10 business class passengers and 225 economy class passengers.

This simplified example assumes all seats are taken and that the same price is charged for all economy seats. In reality there are many more complexities.

Exercise 11

A caterer has been asked to provide two meal options for economy and business class for an airline. The costs of producing the meals, the maximum spend on each meal type and the profit on each type are in the table below:

	Meal 1	Meal 2	Profit
Economy	\$30	\$25	\$70
Business	\$40	\$50	\$100
Max total spend	\$1150	\$1250	

If x is the number of economy meals and y the number of business meals:

- **a** State the constraints.
- **b** State the maximum profit function (objective function).
- **c** Find the number of each type of meal which maximises the profit.
- **d** Is the solution feasible? If not, why not?

Linear programming helps us maximise, or minimise, an objective function, but the solutions generated are not always integers. As we cannot have fractions of a flight, passenger or crew we need to find a way to find integer solutions.

For example, in Exercise 11, the optimal solution did lie at one of the vertices of the feasible region, but gave us a non-integer solution. Here an integer solution is required, since you cannot have part of a meal. The only *feasible optimal* answer would be one of the integer points *within* the feasible region – but which one? The solution can be found by calculating the value for each point manually, but for a complex problem with many variables to control we need a more efficient solution.

Integer programming (IP)

One method of finding integer solutions for simple problems is to test every integer point in the feasible region.

This is where a technique called integer programming is useful.

In Figure 4 we see the feasible region for the linear programming problem given in Exercise 11. The vertex where the lines intersect is the theoretical optimal solution, but

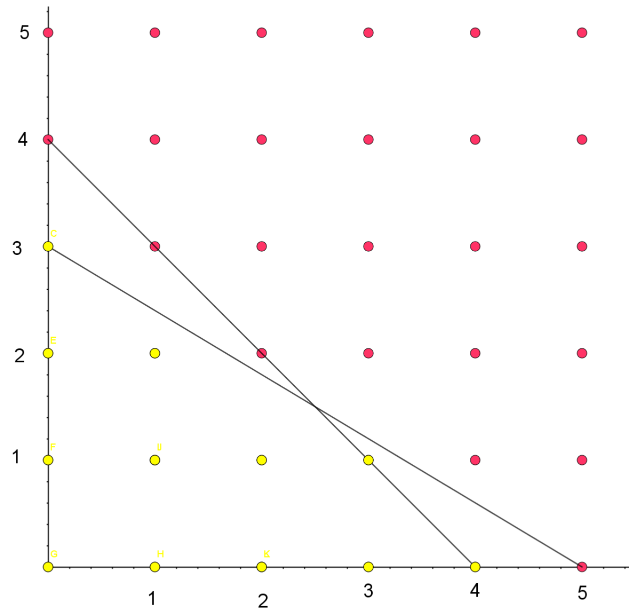


Figure 4: Feasible region for Exercise 11

it does not have integer coordinates so cannot be used. The integer solutions to the problem are in yellow. Each of these would have to be tested with the optimal function to see which one is the optimal solution. With computers able to test each point, this is economically possible for this small problem.

Mathematicians continue to work on finding algorithms to solve integer programming problems where the complexity is too great to check all possible solutions. Integer linear programming is known to belong to a class of problems called "NP-hard". In short, this means there is unlikely to be a 'fast' method which can solve them exactly. Where fast solutions are required, companies must use heuristic methods, which get close to the optimal solution without being guaranteed to be the best.

References

https://www.informs.org/Recognizing-Excellence/INFORMS-Prizes/Daniel-H.-Wagner-Prize-for-Excellence-in-Operations-Research-Practice

http://jlmartin.faculty.ku.edu/ jlmartin/courses/math105-F11/Lectures/chapter5-part2.pdf

http://theconversation.com/why-ups-drivers-dont-turn-left-and-you-probably-shouldnt-either-71432

https://www.youtube.com/watch?v=4oDLMs11Exs

https://www.youtube.com/watch?v=jmCc5VIMOro

Solutions to Exercises

Exercise 1

a Darwin, Brisbane and Melbourne.

b Sydney.

c Three.

Exercise 2

a There are no arcs starting and finishing at the same node.

b The degree of each vertex.

c Darwin has four.

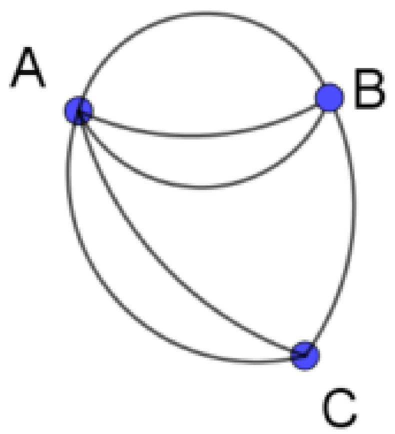
Exercise 3

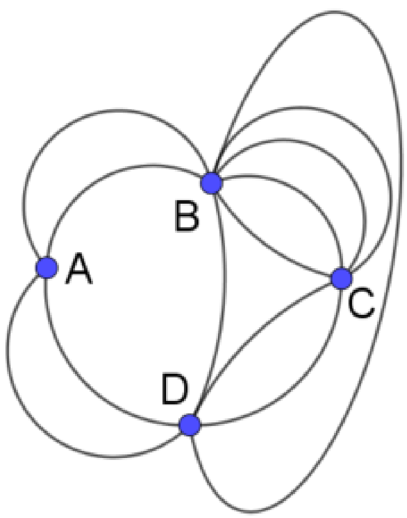
a

0	1	2
1	0	2
2	2	0

0	3	0	1
3	0	2	1
0	2	0	0
1	1	0	0

b





Exercise 4

a

Vertex	Shortest distance from start	Previous vertex
O	**0**	
A	∞, **2**	**O**
B	∞, **4**	**O**
C	∞, **3**	**A**
D	∞, **6**	**C**
F	∞, **6**	**C**
E	∞, 10, **8**	D, **F**

The shortest path is **O – A – C – F – E** with a total distance of 8.

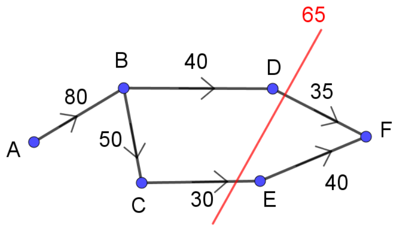
b

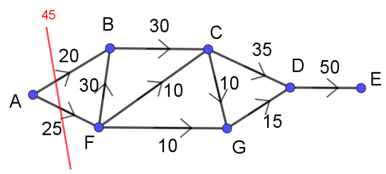
Vertex	Shortest distance from start	Previous vertex
O	**0**	
A	∞, **3**	**O**
B	∞, **2**	**O**
C	∞, **3**	**O**
D	∞, **8**	**C**
F	∞, **8**	**A**
G	∞, **6**	**B**
E	∞, 13, **10**	F, **D**

The shortest path is **O – C – D – E** with a total distance of 10.

Exercise 5

a

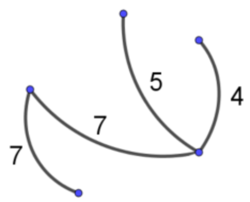




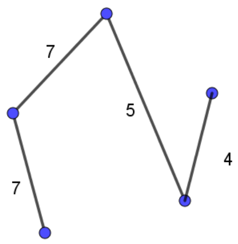
b One solution is to upgrade roads C–E and D–F to each take 40. The capacity would then be 80.

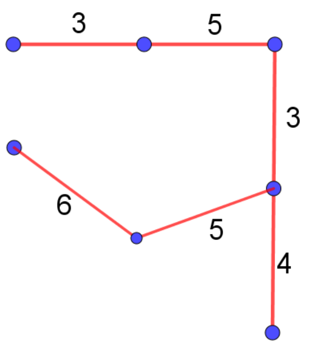
Exercise 6

a



or



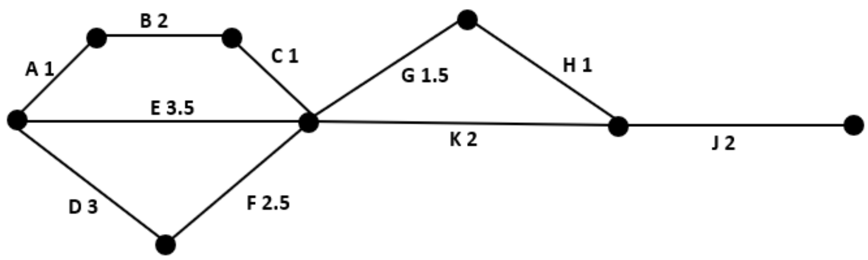


b Either Prim or Kruskal will give these solutions.-

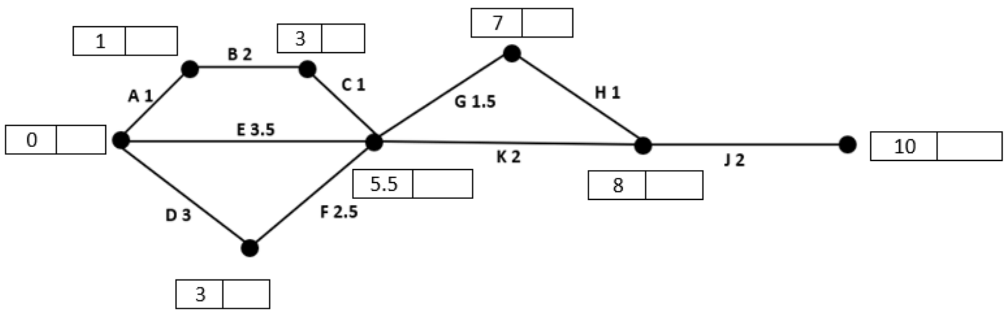
c There are two minimum spanning trees for the first graph.

Exercise 7

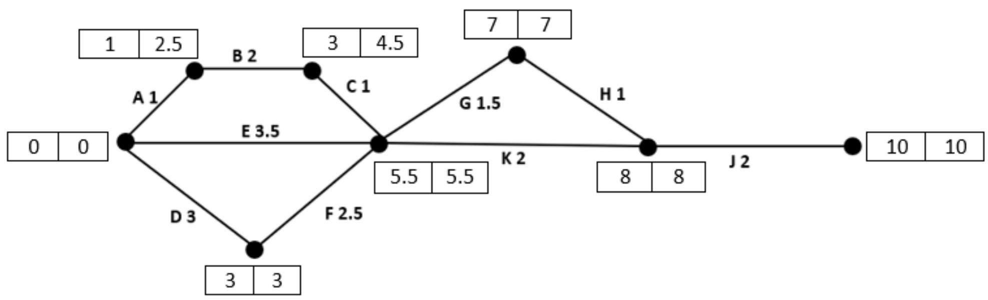
Network Diagram



Earliest Start Time



Latest Start Time



The critical path is where the earliest and latest start times coincide: **D – F – G – H – J**.

To calculate the floats for non-critical activities:

> **Float = LST – EST (previous) – Duration**.

Float(A) = 2.5 – 0 – 1 = 1.5 hours

Float(B) = 4.5 – 1 – 2 = 1.5 hours

Float(C) = 5.5 – 3 – 1 = 1.5 hours

Float(E) = 5.5 – 0 – 3.5 = 2 hours

Float(K) = 8 – 5.5 – 2 = 0.5 hours.

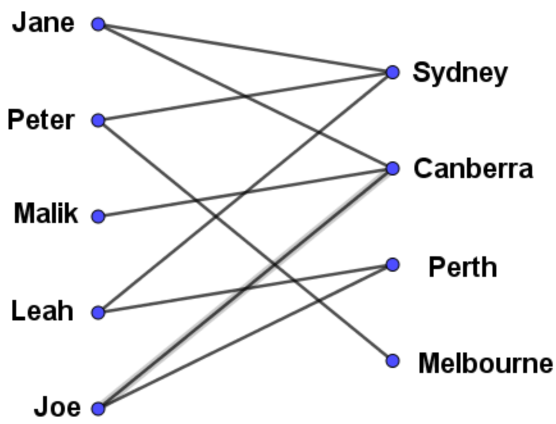
Exercise 8

a Qantas, Virgin Australia and Jetstar.

b Perth, Newman and Brisbane.

c Tigerair.

Exercise 9



Exercise 10

W1	A	70
W2	C	8
W3	B	24
W4	D	35
	Total	137

Exercise 11

a The constraints are

$$x \geqslant 0, \quad y \geqslant 0$$

$$30x + 40y \leqslant 1150 \text{ (constraint on Meal 1)}$$

$$25x + 50y \leqslant 1250 \text{ (constraint on Meal 2)}$$

b The maximum profit function is

$$P(MAX) = 70x + 100y$$

c At $P(MAX)$ we have $x = 15$, $y = 17.5$ and $P = \$2800$.

d This solution is not feasible as you cannot make half a meal!